Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №2

Тема: «Классы и объекты. Инкапсуляция.»

Выполнил

Студент группы РИС-24-1б

Конькова С. С.

Проверил доц. Кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2025

# Постановка задачи

1. Определить пользовательский класс.
2. Определить в классе следующие конструкторы: без параметров, с параметрами, копирования.
3. Определить в классе деструктор.
4. Определить в классе компоненты-функции для просмотра и установки полей данных (селекторы и модификаторы).
5. Написать демонстрационную программу, в которой продемонстрировать все три случая вызова конструктора-копирования, вызов конструктора с параметрами и конструктора без параметров.

# Анализ

# Конструктор без параметров:

# Создается объект p1 без указания начальных значений

# Все поля инициализируются по умолчанию (пустая строка, нулевые значения)

# Конструктор с параметрами:

# Создается объект p2 с явным указанием ФИО, зарплаты и премии

# Поля инициализируются переданными значениями

# Конструктор копирования (три случая):

# Явное копирование: создание p3 как копии p2

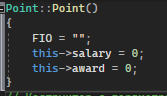
# Передача в функцию по значению: при передаче p2 в лямбда-функцию создается копия

# Возврат из функции по значению: при возврате объекта из лямбда-функции создается копия

# При завершении программы для каждого объекта автоматически вызывается деструктор, о чем будет выведено сообщение.

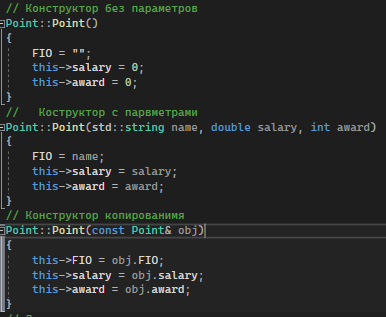
**Контрольные вопросы**

1. Для чего нужен конструктор?

Метод для инициализации объекта класса, который позволяет избежать ошибки, связанные с использованием неинициализированных переменных

- “обнуляем” атрибуты объекта класса Point

1. Сколько типов конструкторов существует

3 типа: конструктор с параметрами и без, конструктор копирования

1. Для чего нужен деструктор? В каких случаях деструктор используется явно?

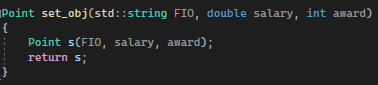
Для удаления ресурсов, выделенных под объект (включая локальные атрибуты). Явно, если создаются динамические типы данных (т.е. указатели), которые нужно очищать.

4-5. Для чего конструктор без параметров? С параметрами? Копирования?

Конструктор копирования вызывается в тех случаях, когда новый объект создается путем копирования существующего:

* при описании нового объекта с инициализацией другим объектом;
* при передаче объекта в функцию по значению;
* при возврате объекта из функции.

Конструктор без параметров нужен для создания “пустого” экземпляра класса.

Конструктор с параметрами для инициализации объекта с заданными параметрами.

1. Перечислите св-ва конструкторов

* Конструктор не возвращает значение, даже типа void. Нельзя получить указатель на конструктор.
* Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).
* Конструктор, вызываемый без параметров, называется конструктором по умолчанию.
* Параметры конструктора могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию, но их может содержать только один из конструкторов.
* Если программист не указал ни одного конструктора, компилятор создает его автоматически. Такой конструктор вызывает конструкторы по умолчанию для полей класса. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.
* Конструкторы не наследуются.
* Конструкторы нельзя описывать с модификаторами const, virtual и static.
* Конструкторы глобальных объектов вызываются до вызова функции main. Локальные объекты создаются, как только становится активной область их действия. Конструктор запускается и при создании временного объекта (на пример, при передаче объекта из функции).

1. Св-ва деструкторов
   * + не имеет аргументов и возвращаемого значения;
     + не наследуется;
     + не может быть объявлен как const или static (далее);
     + может быть виртуальным (далее).
2. К каким атрибутам имеют доступ методы класса

Ко всем элементам класса вне зависимости от спецификатора доступа

1. Что представляет собой запись: this

Указатель на текущий объект

1. Как разница между методами, определенными внутри класса и вне класса

Нет разницы.

1. Какое значение возвращает конструктор?

Конструктор ничего не возвращает

1. Какие методы создаются по умолчанию?

Конструктор без параметров, копирования, деструктор

1. Какое значение возвращает деструктор?

Деструктор ничего не возвращает

Дано описание класса class Student

{

string name; int group; public:

student(string, int); student(const student&)

~student();

};

Какой метод отсутствует в описании класса?

Конструктор без параметров

Какой метод будет вызван при выполнении следующих операторов: student\*s;

s=new student;

Конструктор без параметров

Какой метод будет вызван при выполнении следующих операторов: student s(“Ivanov”,20);

Конструктор с параметрами

Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);

student s2=s1;

Конкструктор с параметрами, копирования

Какие методы будут вызваны при выполнении следующих операторов: student s1(“Ivanov”,20);

student s2; s2=s1;

Конструктор с параметрами и без, копирования

Какой конструктор будет использоваться при передаче параметра в функцию print():

void print(student a)

{

a.show();

}

Конструктор копирования

Класс описан следующим образом:

class Student

{

string name; int age;

public:

void set\_name(string);

void set\_age(int );

…..

};

Student p;

Каким образом можно присвоить новое значение атрибуту name объекта р?

С помощью сеттера p.set\_name(string YOUR TEXT);

UML таблица

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Код программы

Point.h:

#pragma once

#include <string>;

class Point

{

public:

std::string FIO;

Point();

Point(std::string name, double salary, int award);

Point(const Point& obj);

~Point();

double get\_salary();

void set\_salary(double salary);

int get\_award();

void set\_award(int award);

void show();

private:

double salary;

int award;

};

Point.cpp:

#pragma once

#include "Point.h"

#include <iostream>

// Конструктор без параметров

Point::Point()

{

FIO = "";

this->salary = 0;

this->award = 0;

}

// Коструктор с парвметрами

Point::Point(std::string name, double salary, int award)

{

FIO = name;

this->salary = salary;

this->award = award;

}

// Конструктор копированимя

Point::Point(const Point& obj)

{

this->FIO = obj.FIO;

this->salary = obj.salary;

this->award = obj.award;

}

// Деконструктор

Point::~Point()

{

std::cout << "Удаление объекта: " << this << std::endl;

}

// Геттер зарплаты

double Point::get\_salary()

{

return salary;

}

// Сеттер зарплаты

void Point::set\_salary(double salary)

{

this->salary = salary;

}

// Геттер премии

int Point::get\_award()

{

return award;

}

// Сеттер премии

void Point::set\_award(int award)

{

this->award = award;

}

// Вывод локальных атрибутов объекта класса Point;

void Point::show()

{

std::cout << "FIO: " << FIO << std::endl << "salary: " << salary << std::endl << "award: " << award << std::endl;

}

Main.cpp

#include "Point.h"

#include <iostream>

#include <clocale>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

// 1. Демонстрация вызова конструктора без параметров

cout << "1. Конструктор без параметров:" << endl;

Point p1; // вызов конструктора без параметров

p1.show();

cout << endl;

// 2. Демонстрация вызова конструктора с параметрами

cout << "2. Конструктор с параметрами:" << endl;

Point p2("Иванов И.И.", 50000.0, 10); // вызов конструктора с параметрами

p2.show();

cout << endl;

// 3. Демонстрация трех случаев вызова конструктора копирования

// Случай 1: явное создание копии объекта

cout << "3.1 Явное копирование:" << endl;

Point p3(p2); // вызов конструктора копирования

p3.show();

cout << endl;

// Случай 2: передача объекта в функцию по значению

cout << "3.2 Передача в функцию по значению:" << endl;

auto printPoint = [](Point p) { // лямбда-функция, принимающая Point по значению

cout << "Внутри функции: ";

p.show();

};

printPoint(p2); // вызовет конструктор копирования

cout << endl;

// Случай 3: возврат объекта из функции по значению

cout << "3.3 Возврат из функции по значению:" << endl;

auto createCopy = [](Point& orig) -> Point { // лямбда, возвращающая Point по значению

return orig; // вызовет конструктор копирования

};

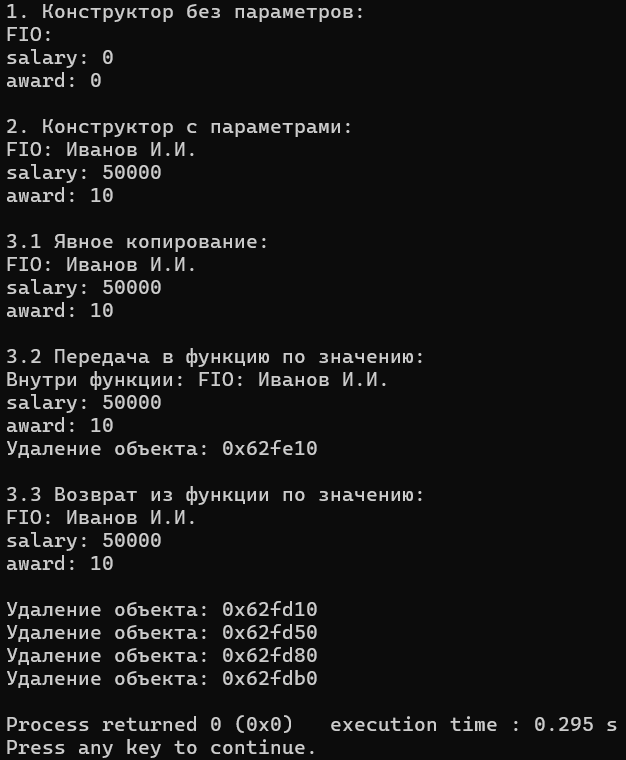
Point p4 = createCopy(p2); // может вызвать конструктор копирования (зависит от оптимизации компилятора)

p4.show();

cout << endl;

return 0;

}

Результат работы: